Using Phase Instead of Optical Flow for Action Recognition

Omar Hommos[1], Silvia L. Pintea[1(✉)], Pascal S. M. Mettes[2], and Jan C. van Gemert[1]

[1] Computer Vision Lab, Delft University of Technology, Delft, Netherlands
Silvia.Laura.Pintea@gmail.com
[2] Intelligent Sensory Interactive Systems, University of Amsterdam, Amsterdam, Netherlands

Abstract. Currently, the most common motion representation for action recognition is optical flow. Optical flow is based on particle tracking which adheres to a Lagrangian perspective on dynamics. In contrast to the Lagrangian perspective, the Eulerian model of dynamics does not track, but describes local changes. For video, an Eulerian phase-based motion representation, using complex steerable filters, has been successfully employed recently for motion magnification and video frame interpolation. Inspired by these previous works, here, we proposes learning Eulerian motion representations in a deep architecture for action recognition. We learn filters in the complex domain in an end-to-end manner. We design these complex filters to resemble complex Gabor filters, typically employed for phase-information extraction. We propose a phase-information extraction module, based on these complex filters, that can be used in any network architecture for extracting Eulerian representations. We experimentally analyze the added value of Eulerian motion representations, as extracted by our proposed phase extraction module, and compare with existing motion representations based on optical flow, on the UCF101 dataset.

Keywords: Motion representation · Phase derivatives
Eulerian motion representation · Action recognition

1 Introduction

Recent advances in action classification rely on training Convolutional Neural Networks (ConvNets) on large video datasets [2,12,20,24]. Such ConvNets need to learn a suitable representation for motion, as it is important for discriminating similar actions that occur in a visually similar context: *e.g. Basketball Throw* and *Basketball Dunk* [11]. Motion information is typically modeled by using optical flow as input to a separate network stream [20], or by using $3D$ convolutions [24], on stacks of input RGB frames and optical flow field stacks [2,5,15,19,20,28]. In this paper, inspired from the suggestions in [18], we zoom in on an alternative

(a) Original input.   (b) Optical flow.   (c) RGB derivative.   (d) Phase derivative.
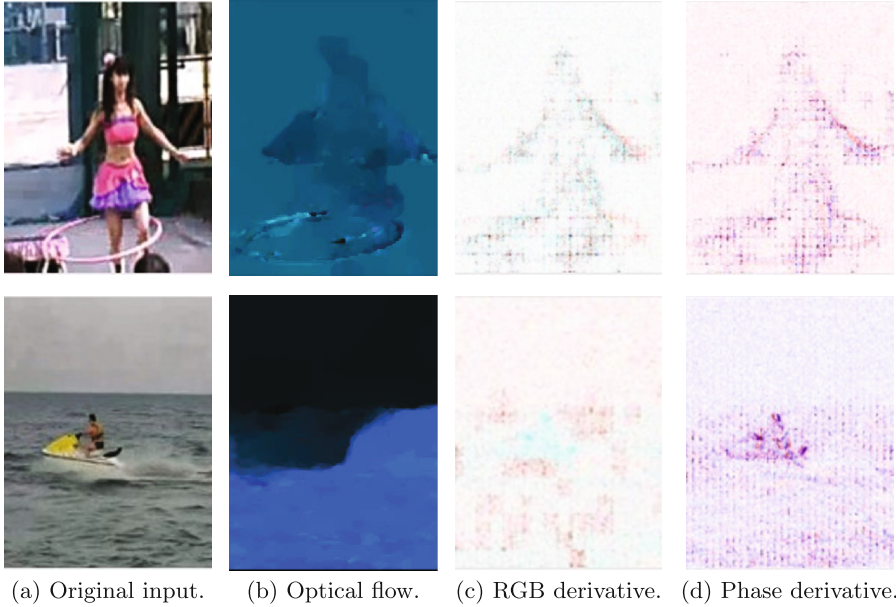
Fig. 1. Motion representations: (a) Original input. (b) Optical flow visualized in HSV space, where hue indicates the direction, and the saturation is the motion magnitude. (c) Difference of RGB (dRGB), over time. (d) Difference of phase (dPhase), over time (as described in Sect. 3). For *dRGB* and *dPhase*, red indicates a positive value, while blue indicates a negative value. The optical flow fails to capture the motion of the waves around the boat. The Eulerian representations better describe the motion at the boundaries of the objects. (Color figure online)

to using optical flow as a motion representation for deep action recognition in video: we investigate using phase in the complex domain instead of optical flow.

Optical flow follows the Lagrangian perspective on motion representations: tracking a pixel over time using its appearance, and the appearance information of neighboring pixels. Dissimilarly, an Eulerian motion representation focuses on the change in image information at a fixed spatial location, over time. Lagrangian methods require explicit point matching to obtain tracks, which is difficult on untextured surfaces, *e.g.* water, hula hoops, or in the presence of occlusion. Eulerian motion representations, instead, do not need to compute explicit correspondences, yet they are sensitive to sudden large motions. Figure 1 depicts this difference between optical flow and Eulerian representations, defined as RGB and phase derivatives over time, on a few video examples.

Previously, Eulerian representations based on RGB differences, have been considered in [27]. However, they were only used as an input to the same network architecture employed for RGB inputs. The performance is highly dependent on the choice of network architecture [2]. Hence, architecture-level changes are necessary to make full use of the Eulerian motion information. In this work, we

propose a phase-extraction module composed of a complex convolutional layer followed by an arctangent function. The proposed module can be trained end-to-end and can be integrated in any existing network architecture.

This work brings forth: (i) the use of Eulerian motion representations for action recognition; (ii) learning phased information in an end-to-end manner by using convolutional layers in the complex domain and complex activations; (iii) an empirical analysis of the advantages and failure cases for the phase-based motion representations, as well as a comparison with existing optical flow-based motion representations on the UCF101 dataset.

2   Related Work

Learning Action Recognition. Top performing action recognition architectures use two-stream networks [4,5,20]. When using a separate motion stream, the input is typically optical flow, and the stream uses popular architectures such as VGG-16 or ResNet, and inter-stream fusion [4,5]. In this work, we adapt the motion stream of [20] architecture for our final network. However, our motion stream describes Eulerian motion.

A natural extension to video is the use of $3D$ convolutions proposed in [24]. In [2], the Inception architecture [22] enhanced with $3D$ convolutions proved effective. This architecture, coined *I3D*, coupled with a new large action recognition dataset, Kinetics [12], delivered state-of-the-art performance. In [28] $3D$ convolutions using $k_t \times k \times k$ filters, are replaced with the more effective $1 \times k \times k$ followed by $k_t \times 1 \times 1$ filters, to reduce computational costs. In this work, we do not consider $3D$ convolutions, but rather focus on learning Eulerian motion descriptions.

In [25] long term convolutions based on optical flow are proposed for action recognition. Recurrent Neural Networks (RNN) are also successfully used for modelling temporal information [3,16]. Our proposed model can be used in combination with such architectures. Where our proposed complex layer plays the feature extraction role.

Motion Representations for Action Recognition. To increase inference speed, in [27] RGB differences are used instead of optical flow. It does not outperform optical flow, yet it obtains comparable performance while being $\times 25$ faster at inference-time. The use of motion vectors, that are similar to optical flow but capture only coarse-motion, is proposed in [29]. The motion vectors obtain a $\times 27$ speed improvement over two-stream networks [20], while having comparable performance. Unlike these methods, we propose learning Eulerian motion representation in an end-to-end framework for tackling action recognition where optical flow fails.

A cascade of networks that learn to generate optical flow for the task of action recognition is used in [15,19]. A similar motion representation is used in [31] before the classification network. Explicit use of optical flow as an input remains superior over other methods for action recognition. However, our aim

here is to research if we can find a complementary motion representation to the optical flow.

Phase-Based Methods. Video phase information has been successfully used before for tasks such as motion magnification and video frame interpolation. Work such as [6,9] reconstruct optical flow by extracting phase changes from image sequences and optimizing for velocity. Measuring phase difference of a stereo image pair helps in estimating disparity at each pixel location for depth estimation [7]. In [13,26,30] motion magnification of small movements was possible by measuring phase variations using a complex steerable pyramids over a sequence of images, and then magnifying motion in a reconstructed video. In [17], a ConvNet architecture was proposed to encode, manipulate, and then decode two subsequent input frames, to obtain an output frame with magnified motion. PhaseNet was proposed in [14], which is a decoder network that receives the decomposition of two input frames, the result of applying steerable pyramid filters, and tries to predict the decomposition of the target frame. Similar to these works, and inspired by [18] we adapt the idea of phase-based motion measurement, however dissimilar to previous works we propose to learn this end-to-end, through complex convolutions.

3 Phase-Based Motion Description

Fourier's shift theorem states that a shift in time domain corresponds to a related linear phase shift in the frequency domain. Since phase variations directly correspond to change [8,17,26], they can serve as a viable representation of motion. This does not only apply to the phase of the Fourier basis functions (sine waves), but also to the phase of other representations in the complex domain such as complex-steerable pyramids [26]. Here, we build on the idea of representing local motion through the phase responses obtained by using a complete set of complex filters.

3.1 Complex Filters for Motion Description

Fleet and Jepson [6] showed that the temporal evolution of contours of constant phase provides a good approximation to the motion field. In [6,9] complex quadrature filters are used to extract the contours of constant phase. The temporal derivative of these responses is then employed to estimate object velocity in videos. Specifically, a set of complex Gabor quadrature filters are used for extracting phase information. A complex quadrature Gabor filter, $H(x, y; \lambda, \theta, \psi, \sigma, \gamma)$, is defined as:

$$H(x, y; \lambda, \theta, \psi, \sigma, \gamma) = G(x, y; \lambda, \theta, \psi, \sigma, \gamma) + iG(x, y; \lambda, \theta, \frac{\pi}{2} - \psi, \sigma, \gamma) \quad (1)$$

$$= \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \times \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right), \quad (2)$$
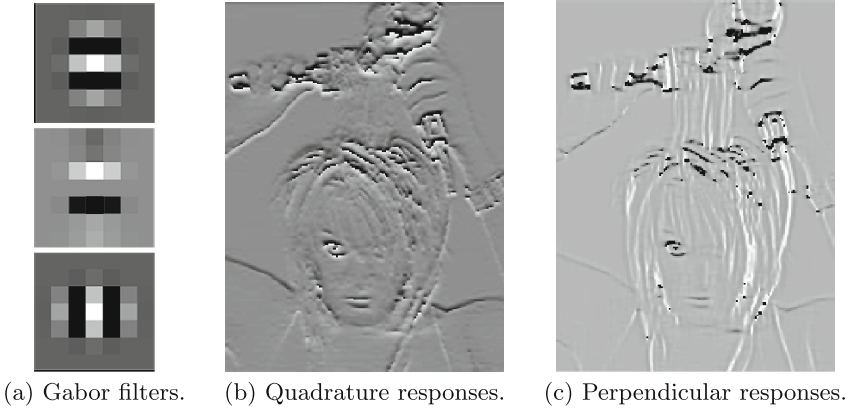
(a) Gabor filters.　(b) Quadrature responses.　(c) Perpendicular responses.

Fig. 2. Quadrauture versus perpendicular filters. (a) From top to bottom: the real and imaginary parts of a complex Gabor, followed by rotated real Gabor. Top and middle form the complex quadrature pair, while top and bottom form a complex perpendicular pair. (b) Phase responses to a complex quadrature Gabor. (c) Phase response to a complex perpendicular filter. In (b) only vertical orientations are highlighted, while both horizontal and vertical orientations are highlighted in (c), providing a more detailed description of the phase contours.

where $G(\cdot)$ is a standard Gabor filter, $i = \sqrt{-1}$, $x' = x\cos(\theta) + y\sin(\theta)$, and $y' = -x\sin(\theta) + y\cos(\theta)$, λ is the wavelength of the wave, θ represents its orientation, ψ is its phase offset, σ is the standard deviation of the Gaussian envelope, and γ is the spatial aspect ratio of the Gaussian, used to control its ellipticity.

In our case, we to not wish to precisely estimate the velocity of objects over time in the video, but rather describe the motion. Therefore, we relax the need of using quadrature filters for finding contours of constant phase and instead, we opt for the more simple perpendicular complex filters.

Perpendicular Filters. Learning complex quadrature filters using convolutional networks is difficult as it requires regularizing to ensure that the phase shift between the real and imaginary filters is $\pi/2$. Here we opt for using perpendicular filters. Combining information from two perpendicular orientations gives a more complete, but less-orientation sensitive, response.

To ensure that the learned filters are perpendicular, we fix the real filters, so they do not receive gradients during the training. We only update the imaginary filters during training and reinitialize the real filters as a $\pi/2$ rotated version of the imaginary filters. This choice avoids numeric problems when extracting the phase information as $\text{atan}(\frac{\mathbf{x}_i}{\mathbf{x}_r})$, where $\mathbf{x}_r$ and $\mathbf{x}_i$ are the real and imaginary responses, respectively.

We find the perpendicular filters to be sufficient for describing motion contours in phase domain, and we validate this in our experiments. Figure 2 depicts the difference between the responses of quadrature complex filters and perpendicular complex filters.
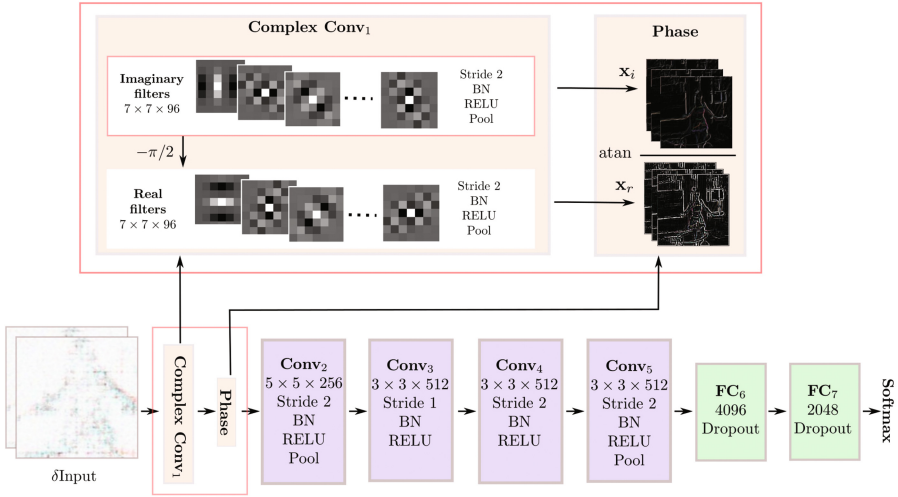
Fig. 3. Network Architecture. We input to our network temporal image derivatives. From the input information, we learn perpendicular complex Gabor-like filters, in our proposed complex layer. We use the responses of these filters to compute the phase. This information is subsequently send to the following layers in the network.

Sinusoidal Gabor Regularization. To encourage the learned filters to resemble Gabor filters, we propose a regularization. We only train the imaginary part of our complex filters. Since the imaginary Gabor filter is a sine multiplied by a Gaussian, we define the imaginary part of our complex filters as a multiplication between a filter initialized randomly and a non-trainable Gaussian kernel. We subsequently, regularize the trainable part of the filter to correspond to a sine.

In the Fourier domain a sine corresponds a single point, ignoring the domain symmetry. Thus, we minimize the L_2 distance from each point of the filter in the Fourier domain to the center of mass of the filter in the Fourier domain:

$$R(\mathbf{w}) = \sum_{i=1}^{n} \|\mathbf{w}_i - \text{CoM}(\mathbf{w})\|_2, \tag{3}$$

where n is the dimensionality of $\mathbf{w}$, $\mathbf{w}$ are the responses of the imaginary filters passed through a Real-FFT (Real Fast Fourier Transform), and $\text{CoM}(\cdot)$ computes the center of mass.

3.2   Learning Phase-Based Descriptions

We adapt a standard ConvNet to learn phase, by learning complex Gabor-like filters. The trainable part of the filters is first initialized randomly. Only the imaginary part of the filters is trained, and regularized with the proposed Gabor regularization. The real part of the complex filter is a $\pi/2$ rotated version of the

imaginary filters. We use these learned filters to extract phase information, and we pass this information to the following layers for action recognition.

Given that the temporal gradient of the phase is the one encoding the motion [6], we need to estimate temporal derivatives of the phase in our network. However, differentiating the responses of a convolution is identical to differentiating one of the functions and then performing the convolution:

$$\frac{\partial}{\partial t}(f * g) = \frac{\partial f}{\partial t} * g, \tag{4}$$

provided that two conditions hold: both functions f and g must be absolutely integrable, and f must have an absolutely integrable (L^1) weak derivative [1]. Given this property, we input temporal image derivatives into our network, to estimated temporal derivatives of phase in our proposed module.

Figure 3 displays our proposed network architecture: we input image derivatives, and from these we learn perpendicular Gabor-like complex filters. We apply the complex non-linearity, $\mathbb{C}$ReLU, proposed in [23] after our complex convolutional layer. $\mathbb{C}$ReLU effectively applies ReLU separately on the real and imaginary feature maps. We also use standard BN (Batch Normalization). We subsequently, estimate the phase as the arctangent of the responses of these filters, and we send this information to the following layers.

4   Experiments

4.1   Experimental Setup

We use the network architecture displayed in Fig. 3. This is a replica of VGG-M, corresponding to one stream in [20], but in which the first layer is replaced with our complex layer. For clarity we will refer to it as: *PhaseStream*. All experiments are performed on UCF101 [21], containing 101 action classes, with an average of 180 frames/video. We follow the standard training/testing data splitting. For the **Exp 1**, where we analyze design choices, we evaluate using only one standard data split. While, for **Exp 2**, we evaluate using the three standard data splits on UCF101. For all experiments, we use momentum SGD as an optimizer with momentum of 0.9. Videos are uniformly sampled from all classes to create a batch of 256. The dropout ratio is set to 0.9 and the learning rate is set to 0.01 and reduced by a factor of 10 at iterations 45000 and 75000. We train for 100,000 iterations. Data is augmented with random crops and flips.

4.2   Exp 1: Importance of Eulerian Information

Exp 1(a): The Use of Perpendicular Versus Quadrature Filters. To quantify the quality of learned perpendicular filters, we initialize our proposed complex layer with rotated Gabor filters. These filters are fixed throughout training. The results serve as a benchmark for the learning process. To compare quadrature filters with the perpendicular ones, our complex layer is initialized with quadrature Gabor filters, also fixed throughout training.
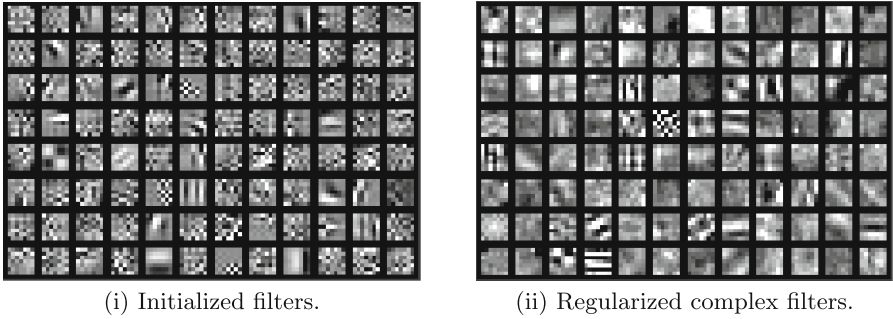
(i) Initialized filters.   (ii) Regularized complex filters.

Fig. 4. Exp 1(a): The effect of the sinusoidal Gabor regularization: (i) randomly initialized filters; (ii) trained filters with the sinusoidal Gabor regularization. The Gabor regularization compels the learned filters to be more similar to Gabor filters.

Table 1. Exp 1(a): Accuracy on UCF101 when using the proposed perpendicular complex filters when compared to quadrature Gabor filters. The network receives as input grayscale image derivatives over time, *dGray*. The perpendicular filters tend to generalize better than the quadrature filters.

Filter type	No. of filters	Training acc	Testing acc
Quadrature Gabor	24	$\sim$90 %	60.5 %
Perpendicular Gabor	24	$\sim$70 %	64.8 %
Perpendicular Gabor	96	$\sim$80 %	71.6 %

Table 1 shows the results of the experiment. We consider two settings for the filter banks: one consisting of 24 filters, similar to the one in [6]; and the other consisting of 96 filters, covering 12 logarithmically spaced frequencies between 0.2 and 5 Hz, over the same 8 directions $\theta = \pi/8 \times \{0, 1, .., 7\}$. From the experimental analysis we conclude that the perpendicular filters help the network generalize better to unseen data.

Figure 4(i) shows the initial set of complex filters, with random initialization. While Fig. 4(ii) shows the learned filters in our complex layer with the incorporated sinusoidal Gabor regularization described in Sect. 3.2. The regularization is effective in encouraging the learned complex filters to resemble Gabor filters.

Exp 1(b): The Importance of the Input. Table 2 shows the performance of different inputs on two network architectures: the VGG-M [20] and our variant of VGG-M in which we replace the first convolution with a complex convolution. We refer to it as *PhaseStream*. In Table 2(i) we consider Eulerian inputs: *dRGB* – derivative of RGB frames obtained by temporally subtracting 2 consecutive frames, *dPhase* – derivative of phase frames, *dGray* – derivative of grayscale frames; and non-Eulerian inputs: *RGB* and OF (Optical Flow). In Table 2(ii) we consider stacked variants of the inputs: *5 × OF* where we stack 5 consecutive OF inputs, *5 × dGray* and *5 × dPhase*. The Eulerian inputs perform better than

Table 2. Exp 1(b): Accuracy (%) on UCF101 for different network inputs for our proposed *PhaseStream* compared to the VGG-M [20]. (i) We compare Eulerian representation: *dGray* – derivative of grayscale inputs over time, *dRGB* – derivative of RGB frames, *dPhase* – derivative of phase images, with *OF* – optical flow, and RGB. The *dPhase* and *dGray* are stronger than using OF as input. (ii) We consider also stacked inputs: $5 \times OF$ – OF stacked over 5 frames, $5 \times dGray$ – 5 grayscale derivatives stacked, and $5 \times dPhase$ – 5 phase derivatives stacked. When stacking OF, $5 \times OF$, there is a substantial gain over stacked Eulerian inputs. Overall, using the proposed *PhaseStream* is beneficial for all inputs except for stacked *dPhase*, $5 \times dPhase$. We highlight in bold the network architecture with the highest accuracy.

Input	VGG-M [20]	PhaseStream (our)
RGB	**52.3** %	51.3 %
OF	67.7 %	N/A
dRGB	45.5 %	**48.8** %
dGray	74.3 %	**74.4** %
dPhase	65.4 %	**70.1** %

(i) Different inputs.

Input	VGG-M [20]	PhaseStream (our)
5×OF	80.4 %	N/A
5×dGray	68.7 %	**75.3** %
5×dPhase	**70.8** %	68.2 %

(ii) Stacked inputs.

the non-Eulerian ones on the VGG-M. However, when stacking the inputs, OF outperforms the rest. Our *PhaseStream* obtains improved performance for all inputs except for $5 \times dPhase$. We do not evaluate our *PhaseStream* network on OF inputs, as computing complex responses over OF does not seem theoretically informative. The stacked Eulerian representations do not perform well due to the large or fast motion, which may result in combining different motion patterns of different object or adding noise into the motion representation.

Table 3 shows the relative improvements in accuracy on the UCF101 dataset for VGG-M [20] and our *PhaseStream* with grayscale frame derivatives, *dGray*, as input. We show the top 10 classes with the largest improvements in accuracy. VGG-M performs better on action categories involving subtle motion, but which are more visual – containing a specific visual object such as: *Archery*, *ApplyLipstick*, or *PlayingFlute*. The *PhaseStream* obtains larger improvements over VGG-M from repetitive activities such as: *BoxingPunchingBag*, *Walking-WithDog*, *MopppingFloor*, *ShavingBeard*. Examples of video frames from these categories are displayed in Fig. 5. We show the network inputs, *dGray* inputs, on the second row. On the last row we show the associated temporal derivatives of phase information, which is the type of information we would expect the network to rely on.

Exp 1(c): Robustness of Motion Information. To quantify the robustness of motion information, we shuffle the input frames before calculating the temporal frame derivatives and OF, to be input to the network. This step effectively removes the temporal structure of the original video [19,28]. We analyze how our proposed *PhaseStream* performs when compared with the VGG-M [20] and

Table 3. Exp 1(b): The relative improvements in accuracy on UCF101 between VGG-M [20] and our *PhaseStream* with *dGray* as input. We show the top 10 classes for each one of the architectures. The networks learn complementary information.

VGG-M [20]	$\Delta\%$	PhaseStream (our)	$\Delta\%$
Archery	19.5 %	FloorGymnastics	19.4%
JumpingJack	15.8 %	TennisSwing	16.3 %
Rowing	13.9 %	BoxingPunchingBag	14.3 %
CricketShot	12.2 %	WalkingWithDog	13.9 %
Skijet	10.7 %	GolfSwing	12.8 %
BlowDryHair	10.5 %	MoppingFloor	11.8 %
PlayingFlute	10.4 %	HighJump	10.8 %
ApplyLipstick	9.4 %	UnevenBars	10.7 %
PlayingCello	9.1 %	ShavingBeard	9.3 %
HulaHoop	8.8 %	HandstandWalking	8.8 %

Table 4. Exp 1(c): Test accuracies on UCF101 when training our *PhaseStream*, VGG-M [20] and TSN [27] on standard inputs, as well as on temporally shuffled inputs. We also show the relative drop in performance, in percentages. The VGG-M suffers a slightly larger relative drop in performance, while TSN and our proposed *PhaseStream* suffer a comparable drop in performance, when the temporal ordering is lost.

Architecture input	VGG-M [20] dGray	TSN [27] $5 \times$ OF	PhaseStream (our) dGray
Standard accuracy	74.3 %	86.9 %	74.4 %
Shuffled accuracy	49.4 %	59.6 %	51.4 %
Relative change (%)	33.5 %	31.4 %	30.1 %

TSN [27]. For our *PhaseStream* as well as for VGG-M, we use as input grayscale temporal image derivatives, *dGray*.

Table 4 shows the test accuracies on UCF101, when feeding the networks standard inputs as well as temporally shuffled inputs. Optical flow numbers are taken from [19] for the TSN architecture [27]. We also show the relative drop in performance, in percentages, caused by the loss of temporal ordering. The TSN performs the best in terms of absolute accuracy scores. When looking at the relative scores, the VGG-M has a slightly larger relative loss of performance, while TSN and our proposed *PhaseStream* suffer a similar relative loss in accuracy.

4.3   Exp 2: Comparison with Existing Work

Table 5 shows the action recognition accuracy on the UCF101 dataset for a number of popular action recognition models: Two-Stream [20], Two-Stream ResNets [4], TSN [27], Motion Vectors [29], ActionFlow [15] and our *PhaseStream*. We
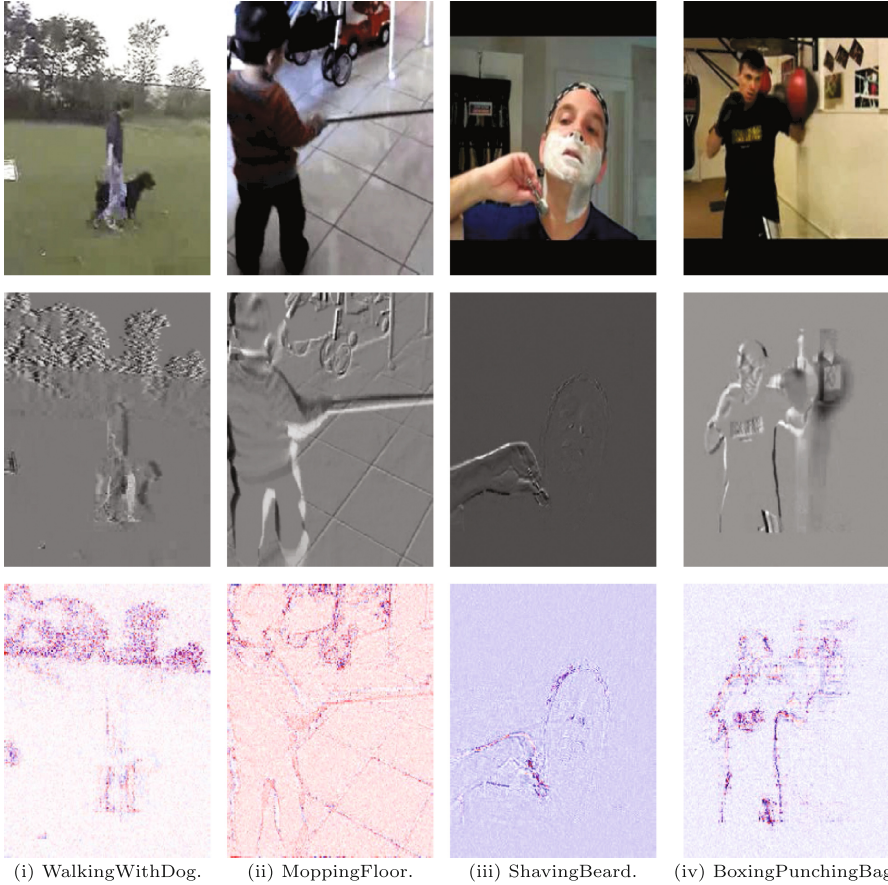
(i) WalkingWithDog.   (ii) MoppingFloor.   (iii) ShavingBeard.   (iv) BoxingPunchingBag.

Fig. 5. Exp 1(b): Examples of video frames from the classes *BoxingPunchingBag*, *WalkingWithDog*, *MopppingFloor*, *ShavingBeard*, where the *PhaseStream* performs better than the VGG-M [20]. We also display the temporal derivatives of grayscale inputs on the second row, and the temporal derivative of phase for these video frames, on the last row. These classes are characterized by repetitive motion patterns.

train our *PhaseStream* on a stack of five differences of grayscale inputs, $5 \times dGray$. Given that we focus on motion representations, we show the performance on the motion (temporal) stream only, for all the considered architectures.

TSN [27] achieves the best performance as it relies on an ensemble of 3 two-stream networks, and provides several architecture-level improvements over them. When it comes to Two-stream network [20], Two-stream ResNets [4], and Motion Vectors [29], the temporal input is a stack of 10 temporal representations. We use only a stack of 5 motion representations, as we did not see a great improvement from temporally stacking the Eulerian representations. Our proposed method on the temporal stream, obtains superior performance to

Table 5. Exp 2: Accuracy (%) on UCF101 comparing our proposed *PhaseStream* trained on $5 \times dGray$, with the Two-stream [20], Two-stream ResNet [4], TSN [27], Motion Vectors [29], and ActionFlow [15]. We show only the accuracy on the motion (temporal) stream for these methods, as we focus only on learning motion representations. The TSN method performs the best. Our proposed approach obtains superior performance to ActionFlow [15], while having slightly lower performance than Motion Vectors [29], and Two-stream ResNets [4], on the temporal stream.

Network	Motion (Temporal) stream
TSN [27]	83.8 %
Two-Stream [20]	81.2 %
Two-Stream ResNets [4]	79.1 %
Motion Vectors [29]	79.3 %
ActionFlow [15]	70.0 %
PhaseStream (our) on $5 \times dGray$	76.4 %

ActionFlow [15], while having slightly lower performance than Motion Vectors [29], and Two-stream ResNets [4]. These results validate that there is gain to be obtained from using Eulerian motion representations for action recognition.

4.4 Limitations and Possible Improvements

Our proposed phase-based motion description shares similar limitations to the classic phase-based approaches, namely dealing with noisy inputs and high-velocity actions. Learning an Eulerian transformation from two consecutive frames independently, could potentially solve the former problem, while increasing the number of proposed complex layers in the network architecture could help the performance.

Another limitation to keep in mind is the number of distinct overlapping motions patterns per spatial neighborhood. Having three or more motion patterns per neighborhood in a video, increases the likelihood of errors in the Eulerian motion representation. This can happen if the effective size of the receptive field of our perpendicular complex filters in the proposed complex layer is too high. A possible improvement to this proposed Eulerian method of learning motion representations for action recognition, is the use of 3D convolutional filters. This may alleviate the problem of achieving limited improvement when stacking Eulerian inputs.

The project was implemented using TensorFlow. The source code for our complex layer can be found at https://github.com/11maxed11/phase-based-action-recognition. More details can be found in [10].

5   Conclusions

We present a new architecture for learning phase-based descriptions from Eulerian inputs, in the context of action recognition. The proposed method relies on learning perpendicular complex filters in a ConvNet. To help the network learn Gabor-like complex filters we propose a regularization scheme based of frequency analysis, for our learned complex filters.

Empirical evaluation shows that this architecture delivers an improvement for several Eulerian inputs, while also exceeding the baseline for recognition using a single optical flow input. Further improvements of the proposed method are possible, by considering different alternative to boost the performance of the temporal representation by using LSTM layers [3,16], and 3D convolutional layers [24].

References

1. Bracewell, R.: "Convolution" and "two-dimensional convolution". In: The Fourier Transform and Its Applications (1965)
2. Carreira, J., Zisserman, A.: Quo vadis, action recognition? A new model and the kinetics dataset. In: CVPR, pp. 4724–4733. IEEE (2017)
3. Donahue, J., et al.: Long-term recurrent convolutional networks for visual recognition and description. In: CVPR, pp. 2625–2634 (2015)
4. Feichtenhofer, C., Pinz, A., Wildes, R.: Spatiotemporal residual networks for video action recognition. In: NIPS, pp. 3468–3476 (2016)
5. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: CVPR, pp. 1933–1941 (2016)
6. Fleet, D.J., Jepson, A.D.: Computation of component image velocity from local phase information. IJCV **5**(1), 77–104 (1990)
7. Fleet, D.J., Jepson, A.D., Jenkin, M.R.: Phase-based disparity measurement. CVGIP Image Underst. **53**(2), 198–210 (1991)
8. Freeman, W.T., Adelson, E.H., et al.: The design and use of steerable filters. TPAMI **13**(9), 891–906 (1991)
9. Gautama, T., Van Hulle, M.M., et al.: A phase-based approach to the estimation of the optical flow field using spatial filtering. TNN **13**(5), 1127–1136 (2002)
10. Hommos, O.: Learning phase-based descriptions for action recognition. Master's thesis, Delft University of Technology, May 2018
11. Jain, M., van Gemert, J.C., Snoek, C.G.: What do 15,000 object categories tell us about classifying and localizing actions? In: CVPR, pp. 46–55 (2015)
12. Kay, W., et al.: The kinetics human action video dataset. CoRR (2017)
13. Kooij, J.F.P., van Gemert, J.C.: Depth-aware motion magnification. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9912, pp. 467–482. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_28
14. Meyer, S., Djelouah, A., McWilliams, B., Sorkine-Hornung, A., Gross, M., Schroers, C.: PhaseNet for video frame interpolation. In: CVPR, pp. 498–507 (2018)
15. Ng, J.Y.H., Choi, J., Neumann, J., Davis, L.S.: ActionFlowNet: learning motion representation for action recognition. CoRR (2016)
16. Ng, J.Y.H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: deep networks for video classification. In: CVPR, pp. 4694–4702. IEEE (2015)

17. Oh, T.H., et al.: Learning-based video motion magnification. CoRR (2018)
18. Pintea, S.L., van Gemert, J.C.: Making a case for learning motion representations with phase. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9915, pp. 55–64. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49409-8_8
19. Sevilla-Lara, L., Liao, Y., Guney, F., Jampani, V., Geiger, A., Black, M.J.: On the integration of optical flow and action recognition. CoRR (2017)
20. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: NIPS, pp. 568–576 (2014)
21. Soomro, K., Zamir, A.R., Shah, M.: UCF101: a dataset of 101 human actions classes from videos in the wild. CoRR (2012)
22. Szegedy, C., et al.: Going deeper with convolutions. In: CVPR, June 2015
23. Trabelsi, C., et al.: Deep complex networks. CoRR (2017)
24. Tran, D., Bourdev, L.D., Fergus, R., Torresani, L., Paluri, M.: C3D: generic features for video analysis. CoRR, abs/1412.0767, vol. 2. no. 7, p. 8 (2014)
25. Varol, G., Laptev, I., Schmid, C.: Long-term temporal convolutions for action recognition. TPAMI **40**(6), 1510–1517 (2018)
26. Wadhwa, N., Rubinstein, M., Durand, F., Freeman, W.T.: Phase-based video motion processing. TOG **32**(4), 80 (2013)
27. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: towards good practices for deep action recognition. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9912, pp. 20–36. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_2
28. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning for video understanding. CoRR (2017)
29. Zhang, B., Wang, L., Wang, Z., Qiao, Y., Wang, H.: Real-time action recognition with enhanced motion vector CNNs. In: CVPR, pp. 2718–2726. IEEE (2016)
30. Zhang, Y., Pintea, S., van Gemert, J.: Video acceleration magnification. In: CVPR. IEEE (2017)
31. Zhu, Y., Lan, Z., Newsam, S., Hauptmann, A.G.: Hidden two-stream convolutional networks for action recognition. CoRR (2017)